

A Big Data Analysis Platform for Healthcare on Apache Spark

Jinwei Zhang, Yong Zhang, Qingcheng Hu, Hongliang Tian, and Chunxiao Xing^(✉)

Tsinghua National Laboratory for Information Science and Technology, Department of Computer Science and Technology, Research Institute of Information Technology,
Tsinghua University, Beijing 100084, China
{jinwei-z15, th112}@mails.tsinghua.edu.cn,
{zhangyong05, huqingcheng, xingcx}@tsinghua.edu.cn

Abstract. In recent years, Data Mining techniques such as classification, clustering, association, regression etc. are widely used in healthcare field to help analyzing and predicting disease and improving the quality and efficiency of medical services. This paper presents a web-based platform for big data analysis of healthcare using Data Mining techniques. The platform consists of three main layers: Apache Spark Layer, Workflow Layer and Web Service Layer. Apache Spark Layer provides basic Apache Spark functionalities as regular Resilient Distributed Datasets (RDD) operations. Meanwhile, this layer provides a cache mechanism to maximize the use of the results as much as possible which were calculated before. Workflow Layer encapsulates a variety of nodes for Data Mining, which have different roles such as data source, algorithm model or evaluation tool. These nodes can be organized into a workflow which is a directed acyclic graph (DAG), and then it will be submitted to Apache Spark Layer to execute. And we have implemented many models including Naïve Bayes model, Decision Tree model and Logistic Regression model etc. for healthcare big data. Web Service Layer implements rich restful API including data uploading, workflow composition and analysis task submission. We also provide a web graphical interface for the user. Through the interface users can achieve efficient Data Mining without any programming which can greatly help the medical staff who don't understand programming to diagnose the patients' condition more accurately and efficiently.

Keywords: Healthcare analysis platform · Cloud computing · Disease prediction · Apache Spark · Big data

This work was supported by the National High-tech R&D Program of China (Grant No. SS2015AA020102), NSFC (91646202), the 1000-Talent program, Tsinghua University Initiative Scientific Research Program.

1 Introduction

In human society, the development of Cloud Computing, Internet of Things and other new services promotes the type and size of data growth at an unprecedented rate. Academia, industry and even government agencies have begun to pay close attention to the big data problems, and more and more people have a strong interest on it. Data Mining is one of the most popular and motivating techniques of discovering meaningful information from varies and huge amount of data. In recent year, Data Mining techniques have been widely used in healthcare field due to its efficient analytical methodology for detecting unknown and valuable information in health data as well as detection of fraud in health insurance, availability of medical solution to the patients at lower cost, detection of causes of diseases and identification of medical treatment methods. It also helps healthcare researchers for making efficient healthcare policies, constructing drug recommendation systems and developing health profiles of individuals [1].

An amount of research works has been done for healthcare using Data Mining techniques. In [2], Data Mining techniques were used to predict Hypertension. And in [3, 4], the authors have utilized classification techniques to predict the likelihood of Cardiovascular Diseases, while in [5, 6], the researchers have put forward integrated Data Mining techniques to detect chronic and physical diseases. Besides, some other researchers [7, 8] developed new methodology and framework for healthcare purpose. However, all of the above-mentioned studies have utilized desktop-based Data Mining techniques. As a result, it's very slow for analyzing the large data.

In the last decade, cloud computing has developed very quickly and provided a new way to establish new healthcare system in a short time with low cost. The "pay for use" pricing model, on-demand computing and ubiquitous network access allow cloud services to be accessible to anyone, anytime, and anywhere [9]. Therefore, it is urgently needed to build an effective big data analysis platform for healthcare which can take advantage of the cloud computing to allow that doctors can quickly and efficiently diagnose the patients' condition. For example, with the help of such platform, doctors can analyze the patient's symptom through the behaviors such as diet, sleep and medical diagnosis history, which will help the doctor gain a more accurate recognition of the patient's condition. Besides, in public health field, the medical big data analysis can also provide healthcare decision-making support, so that patients can get the correct preventive healthcare consulting services and it will change the hospital's medical service model. For medical institutions, the use of big data analysis has become to improve productivity and healthcare services, enhance competitiveness and accelerate economic growth.

Challenges to build such a platform which can handle the large volume of data are as follows:

Challenge I: System Efficiency. As a big data analysis platform, users may not be able to tolerate the long waiting time when handling large data analysis tasks. Therefore, the platform should deal with large data analysis tasks very fast. Besides, for the platform itself, it will take a lot of resources when dealing with the analysis tasks. And of course, we expect that the energy consumption should be meaningful. That is, when a lot of

users execute the analysis tasks in the same time period, if the intermediate results are the same among multiple tasks, the platform should be able to reuse these results as much as possible rather than recalculating these tasks from the beginning, which can avoid unnecessary calculation and save the computing resources.

Challenge II: System Scalability. The massive-scale of available multidimensional data hinders using traditional database management systems. Moreover, large-scale multidimensional data, besides its tremendous storage footprint, makes it extremely difficult to manage and maintain. The underlying database system must be able to digest Petabytes of data and effectively analyze it.

Challenge III: User-Friendliness. User may be a professional, but without good programming ability or even if he or she does not understand programming completely. Therefore, it is important to build a visual interface that users who just need very little knowledge of data mining or machine learning can deal with a big data analysis task.

Existing database systems have a whole set of data types, functions, operators and index structures to handle the multidimensional data operations. Even though such systems provide full support for the data storage and access, they suffer from a scalability issue. Based upon a relational database system, such systems are not scalable enough to handle large scale analytics over big data. The Hadoop distributed computing framework based on MapReduce performs data analytics at scale. The Hadoop-based approach indeed achieves high scalability. However, these systems though exhibit excellent performance in batch-processing jobs, they show poor performance in handling applications that require fast data analysis. Apache Spark, on the other hand, is an in-memory cluster computing system. Spark provides a novel data abstraction called resilient distributed datasets (RDDs) that are collections of objects partitioned across a cluster of machines. Each RDD is built using parallelized transformations (filter, join or group by) that could be traced back to recover the RDD data. In-memory RDDs allow Spark to outperform existing models (MapReduce) by up to two orders of magnitude. In fact, this can meet the demand of fast analysis. Unfortunately, when the same job runs on Spark directly twice, both running cost the same time, that is, the Spark takes the same memory and the same CPU time to execute the same job twice, though it can return the results of the last time directly, which causes the waste of resources.

A cloud based framework [10] has been developed for delivering healthcare as a service. In [11], the work provides a Cloud-based adaptive compression and secure management services for 3D healthcare data which mainly focus on dealing with varies kinds of healthcare data types for further disease detection. The work in [12] provides a cloud based framework for home diagnosis over big medical data. All these research works took the advantages of the fast development, scalability, rapid provisioning, instant elasticity, greater resiliency, rapid re-constitution of services, low-cost disaster recovery and data storage solutions made available by cloud computing [13]. However, these systems are not user-friendly and require professional computer sciences engineers to use.

In this paper, we design and implement a big data analysis platform for healthcare which takes advantages of the mature Data Mining techniques in order to produce precise

results while at the same time harness the opportunities of advanced distributed computing framework—Apache Spark to offer low cost but high quality services. Our platform significantly lowers the required expertise level for coding and machine learning algorithm with the help of web user interface. In the usual big data analysis research, our system has very good performance for the medical data. Our contributions are as follows:

1. We developed a big data analysis system which can convert user-defined machine learning workflow into an Apache Spark job. The workflow consists of different kinds of nodes that can be data source, preprocessing tool, machine learning model and evaluating tool.
2. We design a cache strategy to use the intermediate results calculated before as much as possible, which makes the analysis faster and hardware resource utilization lower. For every node of the user-defined workflow, we calculate the hash code according to the node's content which includes node inputs and node parameters. When a data analysis job is submitted, the system can use the intermediate results calculated by previous job, if the hash code of the nodes which the intermediate results depend on is the same as the new job's corresponding nodes.
3. Our platform provides a web-based graphical user interface to help users build the big data analysis workflow conveniently and quickly. Users just need very little knowledge of data mining or machine learning, that is, it is enough to deal with a big data analysis task for everyone if the one makes sense of the meaning of every node. Those users who are non-specialized programmer don't need to do any coding and can compose a big data analysis job just by clicking and dragging through the mouse in our website.

We demonstrate our platform using an application: we will create a complex workflow including Decision Tree model, Logistic Regression model and Naïve Bayes model to analyze the medical big data which will show the power of quick analyzing. The effectiveness of the cache strategy is proved by modifying part of the whole workflow.

2 Related Work

This section briefly describes data mining, Apache Spark and LRU cache strategy we used in our work.

2.1 Data Mining

Data Mining is an interdisciplinary subfield of computer sciences [14, 15]. It is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Data mining is the analysis step of the “knowledge discovery in databases” process.

The actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown, interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection), and dependencies (association rule mining). In general, such tasks can be classified into two categories: descriptive and predictive. Descriptive mining tasks include association, clustering, and summarization. These tasks, characterize properties of the data in a target data set. Predictive mining tasks include classification, and regression. They perform induction on the current data in order to make predictions [16].

2.2 Apache Spark

Apache Spark provides programmers with an application programming interface centered on a data structure called the resilient distributed dataset (RDD), a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way [17]. It was developed in response to limitations in the MapReduce cluster computing paradigm, which forces a particular linear dataflow structure on distributed programs: MapReduce programs read input data from disk, map a function across the data, reduce the results of the map, and store reduction results on disk. Spark's RDDs function as a working set for distributed programs that offers a (deliberately) restricted form of distributed shared memory [18].

Apache Spark can run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk. Apache Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing. Spark powers a stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming [19].

Spark MLlib is a distributed machine learning framework on top of Spark Core that, due in large part to the distributed memory-based Spark architecture, is as much as nine times as fast as the disk-based implementation used by Apache Mahout and scales better than Vowpal Wabbit. Many common machine learning and statistical algorithms have been implemented and are shipped with MLlib which simplifies large scale machine learning pipelines.

2.3 LRU Cache Strategy

LRU is an abbreviation for Least Recently Used. This cache strategy discards the least recently used items first. And the algorithm requires keeping track of what was used when, which is expensive if one wants to make sure the algorithm always discards the least recently used item. General implementations of this technique require keeping "age bits" for cache-lines and track the "Least Recently Used" cache-line based on age-bits. In such an implementation, every time a cache-line is used, the age of all other cache-lines changes. LRU is actually a family of caching algorithms with members including 2Q by Theodore Johnson and Dennis Shasha [20], and LRU/K by Pat O'Neil, Betty O'Neil and Gerhard Weikum [21].

For CPU caches with large associativity (generally > 4 ways), the implementation cost of LRU becomes prohibitive. In many CPU caches, a scheme that almost always

discards one of the least recently used items is sufficient. So many CPU designers choose a PLRU algorithm which only needs one bit per cache item to work. PLRU typically has a slightly worse miss ratio, has a slightly better latency, uses slightly less power than LRU and lower overheads compared to LRU.

3 Platform Architecture

As depicted in Fig. 1, our platform consists of three main layers: (1) Apache Spark Layer (Sect. 3.1). (2) Workflow Layer (Sect. 3.2). (3) Web Service Layer (Sect. 3.3).

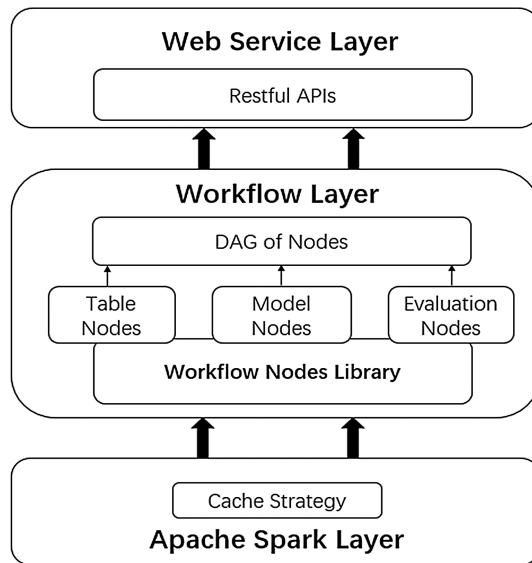


Fig. 1. The architecture of the platform

3.1 Apache Spark Layer

This layer consists of regular operations that are natively supported by Apache Spark and responsible for loading/saving data from/to persistent storage which is either local disk or Hadoop file system HDFS according to the file type and other RDD operations of transformation and action to execute some user-defined workflow. Besides, we design a cache strategy in this layer.

Storage Strategy. For the multidimensional big data needed to analyze, we will generate a scheme file which describe the data type for every column. The DataFrame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood. When we generate a DataFrame object from the multidimensional big data, we need the corresponding scheme file. Because the scheme file

size is very small about a few KB and the HDFS block size is 64 MB at least, so we store the scheme file on the local file system and the data on HDFS.

Cache Strategy. Figure 2 displays a workflow for medical data analysis. Users can click the node of the workflow to view the data or the configuration of the node. We will cache every output of the node and put the intermediate results into a Hash Map. And the key corresponding to the intermediate result is a string of hash code which is calculated according to the input and parameters of the node that is the data and configuration of the node. When the user-defined workflow is executed on the Apache Spark layer, the system will firstly call the cache strategy module to calculate every node's hash code from the origin of the directed acyclic graph and then search the hash code string from the key set of the Hash Map. Once the key is found, the cache module returns the corresponding value immediately. Otherwise, the new key-value pair will be cached. This will save a lot of computing time. Moreover, if the data source is the same as the before, we can save a lot of time wasted in IO (loading data from HDFS).

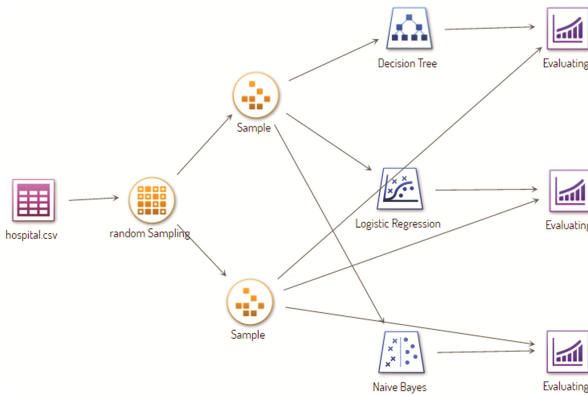


Fig. 2. An example work flow

But memory is limited and the Hash Map will certainly not increase without limit. So we designed an evict policy based LRU algorithm for key-value pairs of the Hash Map. We set the maximum number for key-value pairs in the Hash Map and let a positive integer to indicate the key-value pair's access time. "1" indicates that the key-value pair is accessed just now or a new item. The bigger the integer is; the easier the corresponding key-value pair is evicted from the Map.

On Apache Spark, analysis task of multi-submission using the cache data depends on that they share one Spark Context, that is, there are many jobs belonging to one Spark Application. We set the whole web service as the Spark Application to ensure this. Because the web listening service is a daemon process, the Spark Context will always exist after an analysis task has finished.

3.2 Work Flow Layer

This layer encapsulates a variety of nodes of machine learning workflow, for example, table node that is regard as the data source, model nodes are machine learning algorithms including decision tree node, logistic regression node and so on, evaluation nodes are used to evaluate the results calculated by the model nodes. Besides, it converts the user-defined workflow to a directed acyclic graph and identifies the validity of the DAG. The validation involves two aspects: on the one hand we will verify whether if the workflow is really a DAG, on the other hand we will verify whether if the DAG is legal, that is, whether the nodes meet the correlation between each other. For example, the table node is the data source, so it must be the origin of some DAG and the DAG must be illegal if there is any node as the prefix of the table node.

In fact, we develop a library named workflowLib to allow programmers to develop all kinds of workflow nodes more conveniently. There are mainly two abstract classes in the library: JobNode class and JobNodeParam class. And there is an abstract function named computeResult in the JobNode class. The abstract function needs to define the detail operations for the node, which defines how to deal with the input and get the output. If the programmers want to add a new machine learning model, he or she just needs to define a class which inherits the JobNode class and implement the abstract function computeResult. Accordingly, the programmer needs to implement the JobNodeParam class to define the parameters for the node. The developers don't need to care about how to verify the DAG. They just need to add the new node defined to a workflow.

The user-defined workflow is a JSON file as the input of this layer. According to the detailed definition of every node of the workflow, the workflow layer module will generate a DAG and then submit it to the Apache Spark Layer.

3.3 Web Service Layer

This layer provides the web service to users, which has two functions. On the one hand, users can deal with big data analysis task on our platform without developing the algorithm themselves. For example, they just need to send a JSON file over HTTP request, which will create a workflow on our platform. On the other hand, the whole platform is really a web service running on the Apache Spark as a daemon process that is a Spark Application which guarantees that the Spark Context always exists. This is the base of the cache strategy described in Sect. 2.1.

The web service layer provides rich Restful APIs including upload the data, delete the data, create a workflow, modify a workflow, execute a workflow and so on.

4 Experiment

In this session, we will display our platform using an analytical application which is described below. The data set used in the display is the medical data, which has almost 45000 items and 43 dimensions. We provide a web user interface to be convenient for users to analyze. A screenshot of this tool's main interface "Workflows" is provided in Fig. 3. The tool has a shockingly simple user interface, so that no programming technical

background is needed and you just do your analysis for big data using just drag and drop with the mouse. Figure 3 displays a new blank workflow, where you can customize your own analyzing workflow. On the left side of Fig. 3, there are nodes that are used to create the workflow. The menu of “Tables” in Fig. 3 shows data resources that are uploaded by users through the upload function in “Tables” interface. The menu of “Preparation” in Fig. 3 includes Random Sampling node, Sample node, Filter node, Join node and Select node which are used to preprocess the data source. There are some machine learning algorithm models including Decision Tree node, Logistic Regression node, Naïve Bayes node, Linear Regression node, Gradient Boosting node, K-Means node and Support Vector Machine node in “Models” menu. In the “Evaluating” menu, there are Scoring node and Evaluating node which are used to identify the training model. Figure 4 shows the “Tables Management” interface which provides functions including uploading file, creating folder, deleting file or folder and moving the file, etc. Because there were some projects using our platform, you can see some tables listed in Fig. 4. All uploading files through the uploading function in Fig. 4 will be listed in the “Tables” menu in Fig. 3. User can just drag and drop the nodes listed on the left side of “Workflows” interface into the middle section of Fig. 3. When you drag a node into the workflow, the system will help you combine the dragging node with nodes that have been in the workflow automatically, if one’s output can be the input of the other.

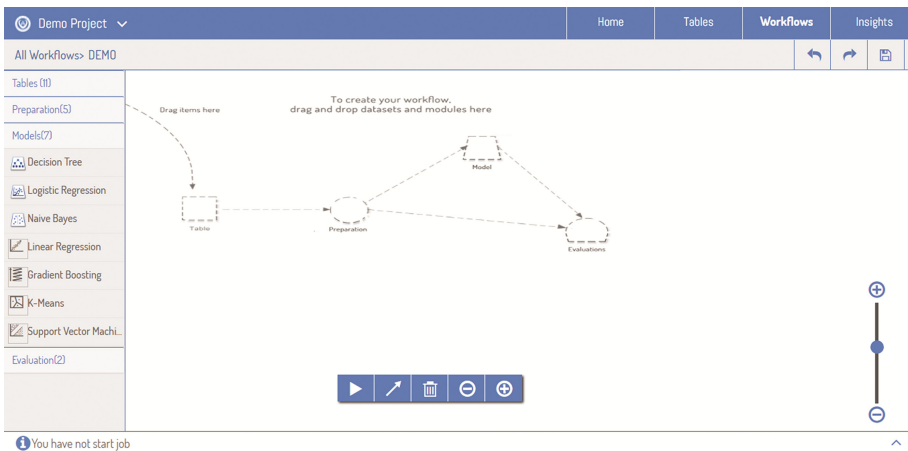


Fig. 3. The main interface of our platform

We will use the workflow displayed in Fig. 2 to demonstrate the big data analyzing. When a user clicks the “Run button” that is a triangle in Fig. 3, all of the nodes will be gray and if one node’s job has been finished, the node will be lighted up. There is an upward arrow in the bottom right corner of Fig. 3 and a user can look over the log information by clicking that arrow. Users can click any node to configure it. For example, the Random Sampling node can define the percent of training set and testing set and the Decision Tree node can configure the training target from the 43 dimensions in the data set and it can also define the parameters for the process of constructing the Decision

Tree which is displayed in Fig. 5. When all of the nodes have lighted up, the whole workflow is completed. Users can click the “Evaluating” nodes to view the results for the big data analysis.

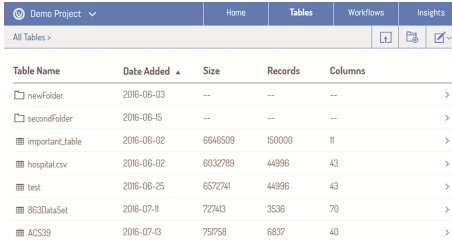


Table Name	Date Added	Size	Records	Columns
newFolder	2016-06-03	--	--	--
secondFolder	2016-06-15	--	--	--
important_Table	2016-06-02	6648509	150000	11
hospital.csv	2016-06-02	6032789	44996	43
test	2016-06-25	6572741	44996	43
863DataSet	2016-07-11	727443	3536	70
ACS33	2016-07-13	75758	6837	40

Fig. 4. The data sources

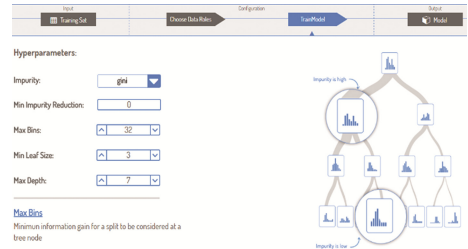


Fig. 5. The result

Because the results of every time for random sampling are different, the input of nodes of the same workflow is different except the table node. In order to present our cache strategy, we divide the table node into two parts manually, and execute the same workflow twice. But there is a little of difference that we change the target of the Decision Tree node. In the first execution, the running time of the Logistic Regression node is more than the Decision Tree node’s obviously. But the opposite occurs in the second execution. Because the input and configuration are the same in two executions for the Logistic Regression node, the results of it should be same. Indeed, that is also the fact, which indicates our cache strategy is correct and effective.

We know that it’s the presupposition and key to analyze the hospital’s electronic medical records data(EMR) for the Clinical Decision Support System. In this paper, we will take the acute chest pain data analysis process as an example to display the excellence of our platform.

Firstly, we selected 1571 data items of cardiovascular patients from the acute chest pain information database, which includes four kinds of cardiovascular disease, that is, aortic dissection, pulmonary embolism, myocardial infarction and steno cardia. The detail information of the data is displayed in Table 1. We extract 31 disease features from patients’ symptoms and signs, history feature, laboratory inspection and so on. And then, we pre-processed the data of qualitative variables and quantitative variables in 31 pathologic features and according to the actual data characteristics we get the final 17 pathologic features to analyze. Finally, we take advantage of the Decision Tree model, Naïve Bayes model and Logistic Regression model to predict the incidence of the four kinds of cardiovascular disease. Figure 6 displays the weights influence of each disease feature on different diseases using Logistic Regression model. And Fig. 7 displays the prediction accuracy of Logistic Regression model. From Fig. 7, we can find that the prediction accuracy for aortic dissection is more than 99%, but just 64% for pulmonary embolism, which is because the amount of data for pulmonary embolism is relatively small and only 78/1571 of the total data.

Table 1. Data of cardiovascular patients.

Disease	Aortic dissection	Pulmonary embolism	Myocardial infarction	Steno cardia
Number	719	78	266	497
Total	1571			

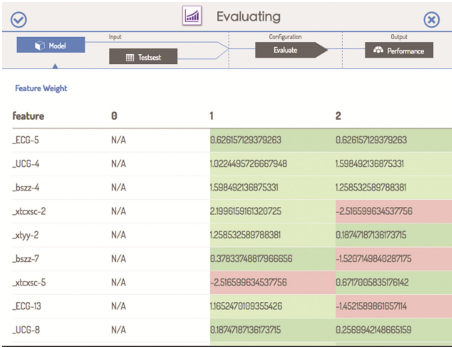


Fig. 6. The weights influence of each disease feature

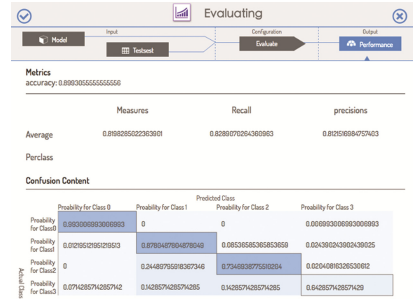


Fig. 7. prediction accuracy of LR model

5 Conclusion

We develop a web service platform on Apache Spark for big data analysis on healthcare data. In our system, we propose a cache strategy to realize the high efficiency of the system, which does not exist in other big data analyzing platform as we know. And in Workflow Layer, we develop a workflowLib which indeed is a framework for implementing user-defined nodes of workflow of machine learning. Besides, we develop a web user interface through which anyone can do the big data analyzing easily. At present, there are some big data analysis tasks running on our platform and it presents a good performance.

In the future, we will develop a stronger platform on which users can write own code for the model nodes in the web user interface when the nodes we provide cannot meet their demands. We will also compile the user-writing code to a model node that will be provided to others directly if the user agrees to publish the code.

References

1. Koh, H.C., Tan, G.: Data mining application in healthcare. *J. Healthcare Inf. Manage.* **19**(2) (2005)
2. Aljumah, A.A., Ahamad, M.G., Siddiqui, M.K.: Predictive analysis on hypertension treatment using data mining approach in Saudi Arabia. *Intell. Inf. Manage.* **3**, 252–261 (2011)
3. Dangare, C.S., Apte, S.S.: Improved study of heart disease prediction system using data mining classification techniques (2012)

4. Kumari, M., Godara, S.: Comparative study of data mining classification methods in cardiovascular disease prediction. *IJCST* **2**(2) (2011). ISSN: 2229- 4333
5. Huang, M.-J., Chen, M.-Y., Lee, S.-C.: Integrating data mining with case-based reasoning for chronic diseases prognosis and diagnosis. *Expert Syst. Appl.* **32**, 856–867 (2007)
6. Ha, S.H., Joo, S.H.: A hybrid data mining method for the medical classification of chest pain. *Int. J. Comput. Inf. Eng.* **4**(1), 33–38 (2010)
7. Amendola, S., Lodato, R., Manzari, S., et al.: RFID technology for IoTbased personal healthcare in smart spaces. *IEEE Internet Things J.* **1**(2), 144–152 (2014)
8. Jung, E.Y., Kim, J., Chung, K.Y., et al.: Mobile healthcare application with EMR interoperability for diabetes patients. *Cluster Comput.* **17**(3), 871–880 (2014)
9. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R.H., Konwinski, A., Lee, G., Patterson, D.A., Rabkin, A., Stoica, I., Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing, UCB/EECS 2009-28, 10 February 2009
10. Kaur, P.D., Chana, I.: Cloud based intelligent system for delivering healthcare as a service. *Comput. Methods Programs Biomed.* **113**(1), 346–359 (2014)
11. Castiglione, A., Pizzolante, R., De Santis, A., et al.: Cloud-based adaptive compression and secure management services for 3D healthcare data. *Future Gener. Comput. Syst.* **43**, 120–134 (2015)
12. Lin, W., Dou, W., Zhou, Z., et al.: A cloud-based framework for Home diagnosis service over big medical data. *J. Syst. Softw.* **102**, 192–206 (2015)
13. Buyya, R., Yeo, C.S., Venugopal, S., Broberg, J., Brandic, I.: Cloud computing and emerging IT platforms: vision hype, and reality for delivering computing as the 5th utility. *Future Gener. Comput. Syst.* **25**(6), 599–616 (2009)
14. Clifton, C.: *Encyclopædia Britannica: Definition of Data Mining* (2010). Accessed 09 Dec 2010
15. Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Accessed 07 Aug 2012
16. Wolford, D.K.: System, pad and method for monitoring a sleeping person to detect an apnea state condition: U.S, Patent Application 12/359,459[P] (2009)
17. Zaharia, M., Chowdhury, M., Franklin, M.J., Shenker, S., Stoica, I.: Spark: cluster computing with working sets (PDF). In: *USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*
18. Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M.J., Shenker, S., Stoica, I.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing (PDF). In: *USENIX Symposium Networked Systems Design and Implementation*
19. Sparks, E., Talwalkar, A.: Spark Meetup: MLbase, distributed machine learning with spark. In: *slideshare.net. Spark User Meetup, San Francisco, California* (2013). [slideshare.net](https://www.slideshare.net/spark), Accessed 10 Feb 2014
20. Johnson, T., Dennis, S.: X3: A Low Overhead High Performance Buffer Management Replacement Algorithm (1994)
21. O’neil, E.J., O’neil, P.E., Weikum, G.: The LRU-K page replacement algorithm for database disk buffering. In: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. SIGMOD 1993*, pp. 297–306. ACM, New York (1993)